Attorney Docket No: HYPE-001/01US

PATENT

Box Patent Application
Commissioner for Patents
Washington, D.C. 20231

UTILITY PATENT APPLICATION TRANSMITTAL

1. Transmitted herewith for filing is a U.S. Non-Provisional Utility Patent Application entitled: **SYSTEM AND METHOD FOR PROVIDING A WEB-BASED OPERATING SYSTEM**

naming as inventors: **Fredrik Malmer**
Erik Viktor Arvidsson
Drew E. Morris
Shervin Pishevar

and including:

[X] (28) pages of description (before the claims);
[X] (6) pages of claims ((39) total claims; (5) independent claims);
[X] (1) Sheet of Abstract; and
[X] (6) sheets of [X] informal []formal drawing(s) including Figures 1-6 .

2. Also enclosed are:
[X] Executed Declaration
[X] Assignment and Assignment Recordation Cover Sheet
[X] Power of Attorney
[X] Statement Claiming Small Entity Status under 37 C.F.R. §§ 1.9 and 1.27
[] Information Disclosure Statement
[] Preliminary Amendment
[] Bibliographic data entry sheet
[] Other:

3. [] Please amend the specification by inserting before the first line the sentence: --This application claims priority under 35 U.S.C. §§119 and/or 365 to ______ filed in ____ on _____, the entire content of which is hereby incorporated by reference.--

4. [] Please amend the specification by inserting before the first line the sentence: --This application claims priority under 35 U.S.C. §119(e) to U.S. Provisional Application No. ________, filed _________, the entire content of which is hereby incorporated by reference.--

5. The filing fee has been calculated as follows [] and in accordance with the enclosed preliminary amendment:

	NO. OF CLAIMS		EXTRA CLAIMS	RATE	FEE
Basic Application Fee					$710.00
Total Claims	39	- 20 =	19	x $18.00	$342.00
Independent Claims	5	- 3 =	2	x $80.00	$160.00
If multiple dependent claims are presented, add $270.00					0
Total Application Fee					$1212.00
If a statement claiming small entity status is enclosed, subtract 50% of Total Application Fee					$606.00
Other fees: (specify)					
TOTAL FEE DUE					$606.00

[ ] This application is being filed without a filing fee.

[ ] Check No. _________ in the amount of $____ for the total fee is attached.

[X] Please charge $606.00 to Deposit Account No. 50-1283 for the total fee.

The Commissioner is hereby authorized to charge any appropriate fees under 37 C.F.R. §§1.16, 1.17, and 1.21 that may be required by this paper, and to credit any overpayment, to Deposit Account No. 50-1283.

6. Please direct all correspondence concerning this application to:

COOLEY GODWARD LLP
Attention: Patent Group
One Freedom Square - Reston Town Center
11951 Freedom Drive
Reston, Virginia 20190-5601
Tel: (703) 456-8000
Fax: (703) 456-8100

CUSTOMER NUMBER: **022903**

Respectfully submitted,
COOLEY GODWARD LLP

Dated: 10/11/2000

By: [illegible]
Frank V. Pietrantonio
Reg. No. 32,289

Cooley Godward LLP
Attention: Patent Group
One Freedom Square - Reston Town Center
11951 Freedom Drive
Reston, Virginia 20190-5601
Tel: (703) 456-8000
Fax: (703) 456-8100

69175 v1/RE
1HDJ01!.DOC

Attorney Docket No: HYPE-001/01US **PATENT**

Applicant or Patentee:	Frederik Malmer *et al.*		
Serial No.:	Not Yet Assigned	Patent No.:	
Filed:	Herewith	Issued:	
For:	SYSTEM AND METHOD FOR PROVIDING A WEB-BASED OPERATING SYSTEM		

VERIFIED STATEMENT CLAIMING SMALL ENTITY STATUS 37 CFR SECTIONS 1.9(f) & 1.27(c) - SMALL BUSINESS CONCERN

I hereby declare that I am:

[] the owner of the small business concern identified below:
[X] an official of the small business concern empowered to act on behalf of the concern identified below:

Name of Small Business Concern:	WebOS, Inc.
Address of Small Business Concern:	10500 Little Patuxent Parkway, Suite 500, Columbia MD 21044

I hereby declare that the above-identified small business concern qualifies as a small business concern as defined in 13 CFR 121.12, and reproduced in 37 CFR 1.9(d), for purposes of paying reduced fees under 35, United States Code 41(a) and (b), in that the number of employees of the concern, including those of its affiliates, does not exceed 500 persons. For purposes of this statement, (1) the number of employees of the business concern is the average over the previous fiscal year of the concern of the persons employed on a full-time, part-time or temporary basis during each of the pay periods of the fiscal year, and (2) concerns are affiliates of each other when either, directly or indirectly, one concern controls or has the power to control the other, or a third party or parties controls or has the power to control both.

I hereby declare that rights under contract or law have been conveyed to and remain with the small business concern identified above with regard to the invention identified above and described in

[X] the specification filed herewith
[ ] the application identified above
[] the patent identified above.

If the rights held by the above-identified small business concern are not exclusive, each individual, concern or organization having rights to the invention is listed below* and no rights to the invention are held by any person, other than the inventor, who could not qualify as an independent inventor under 37 CFR 1.9(c) or by any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e).

*NOTE: Separate verified statements are required from each named person, concern or organization having rights to the invention averring to their status as small entities. (37 CFR 1.27)

Name:
Address:

[ ] Individual [ ] Small Business Concern [ ] Nonprofit Organization

I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b))

I hereby declare that all statements are made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

Signature	[illegible]
Date	9/13/00
Name of Person Signing	Shervin Pishevar
Title of person other than owner	Chief Executive Officer
Address of person signing	10500 Little Patuxent Parkway, Suite 500, Columbia MD 21044

64792 v1/RE
1DZS01! DOC

SYSTEM AND METHOD FOR PROVIDING A WEB-BASED OPERATING SYSTEM

Background

Cross-Reference to Related Applications

The present application claims priority to the following provisional application, which is incorporated herein by reference in its entirety: U.S. Provisional Application No. 60/186,304, entitled "SYSTEM AND METHOD FOR PROVIDING A WEB-BASED OPERATING SYSTEM," filed March 1, 2000.

Field of the Invention

The present invention relates generally to Internet-based computer applications and more particularly to providing these applications to a user from a remote server.

Discussion of the Related Art

In conventional computer systems, a computer application operates within the confines of an operating system resident on a local computer. The operating system provides a layer of abstraction between the computer application and the operation of the physical computer hardware associated with the local computer. The operating system enables developers of the computer application to provide various functionality to a user regardless of the type of physical computer hardware owned by the user. For example, the operating system provides the computer application with access to hard drives, monitors, input/output ports, peripherals, and various other devices without having to understand the nature and operation of the particular piece of physical hardware associated with the local computer.

In conventional computer systems, the computer application is resident on the local computer. When the user wishes to access a particular computer application, the user selects the computer application via an operating system user interface to initiate or run the computer application. For example, in a Windows™ operating system, the user may select the particular computer application by "double-clicking" on an icon associated with the particular computer application from the "desktop" or selecting the particular computer application from a menu such as Windows™ "Program" menu. Once the particular computer application is selected, the operating system retrieves the particular computer application from a storage device at the local computer and runs it within the context of the operating system in a well-known manner.

With conventional computer systems, the user usually has to purchase a copy of each computer application that he intends to install and use on a computer. Often, these computer applications are expensive (i.e., paying for the installation of the application, or the per instance cost). When the application is purchased in this manner, the cost of a computer application on a per use basis is much higher for a casual user than a heavy user. It would be more economically efficient, if users paid for the computer application on a per use basis rather than on a per instance basis. The conventional configuration, however, does not provide a "pay-for-use" mechanism for the computer application.

Distribution is another problem associated with purchasing individual copies of a computer application. In conventional computer systems, computer applications are distributed to users via a software medium such as a floppy diskette. The user loads the computer application from the floppy diskette onto his local computer. Subsequent modifications to the application (e.g., corrections, modifications, additions, etc.) must be delivered to the user via a software medium as well, often times at additional expense to the user. To avoid this expense, many users simply retain an outdated (and often "buggy") version of the computer application. Moreover, developers may delay the release of "bug fixes" for long periods of time with the intent of releasing the "bug fixes" with newly developed features. Meanwhile, users "make do" with a less than perfect computer application.

Yet another problem associated with individual copies of a computer application is the wasteful use of computer storage media. For example, it is not uncommon that a computer application, when fully installed on a local computer, require tens of megabytes of storage space on a computer hard drive. Typically, the amount of storage space required greatly increases with each subsequent release of the computer application. While storage space is becoming cheaper every year (in terms of dollars/megabyte), much of storage space is occupied by code functions never utilized by the user. This problem is compounded by copies of a computer application installed by millions of users, resulting in large scale storage media inefficiencies.

In view of the shortcomings of known techniques for distributing and operating applications programs, it would be useful to have a system and method that provide an alternative technique for delivering an operating system and computer applications.

Summary of the Invention

The present invention is directed to a system and method that solves the problems of conventional computer systems by providing a web-based operating system and web-based computer applications to a user via a network. A web-based operating system is downloaded from a server onto a network-enabled device. The web-based operating system provides a platform from which web-based computer applications are executed. Web-based computer applications are downloaded from a server onto the network-enabled device, and executed in conjunction with the web-based operating system.

According to one feature of the present invention, users access web-based computer applications on demand. When a user wishes to access a web-based computer application, the web-based computer application is downloaded from a server for use at a network-enabled device. The user may access and repeatedly download the web-based computer application as often as needed, rather than pay a high initial price for a conventional computer application. In this manner, the present invention provides a mechanism by which the user may "pay-for-use" of the web-based computer application. This solves the problem of the high initial cost of conventional computer applications.

According to another feature of the present invention, the web-based computer application software is distributed from a server when the user chooses to access the web-based computer application. When a new version of the web-based computer application software becomes available, it is released onto the server. Since the web-based computer application software is downloaded when the user wishes to execute the web-based computer application, the user always has access to the newest version of the software. This solves the problem of distribution of individual copies of a conventional computer application.

According to yet another feature of the present invention, portions of the web-based operating system software and web-based computer application software are downloaded from the server to the network-enabled device as needed. When a user first accesses the web-based operating system, the core software for operation of the web-based operating system are downloaded from the server to the network-enabled device. When the user accesses a web-based computer application, the web-based computer application software and any additional web-based operating system software needed are downloaded from the server. This results in storing only that software at the network-enabled device necessary for executing the web-based computer application. This solves the problem of wasted storage media associated with storing individual copies of conventional computer applications in conventional computer systems. This, and other advantages of the present invention are described below.

Brief Description of the Drawings

The foregoing and other features and advantages of the invention will be apparent from the following, more particular description of a preferred embodiment of the invention, as illustrated in the accompanying drawings.

FIG. 1 illustrates a computer network that may be used to implement the present invention.

FIG. 2 illustrates a computer system for providing a server-side computer application according to one embodiment of the present invention.

FIG. 3 illustrates an exemplary embodiment of the present invention.

FIG. 4 illustrates the process of running a web-based application according to the present invention.

FIG. 5 illustrates a file according to the present invention.

FIG. 6 illustrates the process of loading a file according to the present invention.

Detailed Description

A preferred embodiment of the invention is discussed in detail below. While specific implementations are discussed, it should be understood that this is done for illustrative purposes only. A person skilled in the relevant art will recognize that other components and configurations may be used without parting from the spirit and scope of the invention.

The present invention is directed to a system and method for providing a web-based operating system and web-based applications to a user via a network. Generally, the present invention eliminates the need for storing a copy of a computer application on a local computer. Rather, the computer application resides on a server. When a user desires to access the computer application, the user selects the application from a desktop that is managed and operated from the server. The computer application (or various aspects, features, or functions thereof) is delivered to the local computer from the server on an "as needed" basis. Such a computer application is referred to as a "server-side" computer application whereas a conventional computer application residing on the local computer is referred to as a "client-side" computer application.

In one embodiment, the present invention provides a web-based operating system that acts as an interface layer between the server-side computer application and the local computer. More particularly, the present invention provides an interface between the server-side computer application and a web-browsing engine operating on the local computer. While described in these terms for purposes of clarity, it will become apparent to those skilled in the art how various aspects of the present invention may be applied to other embodiments.

The present invention provides a component-based environment for the development and deployment of a web-based operating system and web-based applications. Components are small programs, or objects, that perform specific functions and are designed in such a way as to easily operate with other components and applications. The present invention deploys the components within library files as part of a component-based development model. The web-based operating system only downloads those library files that include components needed to execute a particular web-based application a user is accessing. Once downloaded, the components may be reused for other web-based applications when needed.

The component-based development model of the present invention provides a number of benefits. First, the component-based development model provides for faster web-based computer application and web-based operating system development. Components need only be developed once, and thereafter reused as building blocks for many applications. Web-based computer applications developed using the component-based development model are more robust because they use code that is tested, and known to work. Additionally, the present invention provides for loading component libraries once, and referencing them from the web-based computer applications many times. This allows many applications to reference components that were downloaded only once, thereby decreasing the number of downloads and increasing the speed of the web-based operating system.

FIG. 1 illustrates a computer network 100 including a user 110 operating a network-enabled device 120 connected to at least one server 130 via a network 140. Network-enabled device 120 may be any electronic communicating device capable of interfacing with network 140. Such devices may include computers, laptops, telephones, cellular phones, personal data accessories ("PDA"), pagers, web enabled televisions ("WebTV"), or other similar electronic communication devices, as would be apparent. Network 140 may be any form of interconnecting network including an intranet, such as a local or wide area network, or an extranet, such as the World Wide Web, or the Internet. Such networks 140 may include various wireless connections as would be apparent. Server 130 may be any sort of storage device for providing web-related data to user 110 as would be apparent.

Conventionally, user 110 accesses a particular web page using a web browser (also referred to as web-browsing engine) such as Microsoft's Internet Explorer™ or Netscape's Navigator™ operating on network-enabled device 120. In order to access a web page, the web browser sends a request to a particular web site using a Uniform Resource Locator ("URL") address associated with the desired web page. The request is passed through network 140 using an appropriate network protocol. For example, when network 140 comprises the Internet, a Hyper-Text Transfer Protocol ("HTTP") is used that encapsulates the request to facilitate its transmission through network 140. Other network protocols may also be used as would be apparent.

Using the URL address, the request is routed to a server (or servers) 130 hosting the web page. Once located, the appropriate server 130 analyzes the request and sends web-related data corresponding to the request back to network-enabled device 120 using the appropriate network protocol (*e.g.*, HTTP). The browser receives the web-related data at network-enabled device 120. This process of requesting and retrieving web-related data is well known.

FIG. 2 illustrates the layers of hardware and software included by network-enabled device 120 for providing a server-side computer application to user 110 according to one embodiment of the present invention. Network-enabled device 120 includes a computer platform, or computer hardware, 210. Operating and residing locally on computer hardware 210 is computer operating system 220. Some examples of well-known computer operating systems

220 are Microsoft's Windows 98™, Windows CE™, and Palm OS™. Computer operating system 220 may be any software, firmware or other program logic that provides an interface between the hardware of network-enabled device 120 and a conventional local computer application, as would be apparent.

One such conventional local computer application operating on computer hardware 210 is preferably a web-browsing engine 230, sometimes referred to as a web browser 230. The operation and use of web-browsing engine 230 are generally well known as mentioned above. Web-browsing engine 230 may include for example, Microsoft's Internet Explorer™, version 5.0 or Netscape's Navigator/Communicator™. In a preferred embodiment of the present invention, web-browsing engine 230 is capable of rendering data in the Hypertext Markup Language ("HTML") and Dynamic Hypertext Markup Language ("DHTML") or similar markup language formats, and executing JavaScript code, as is apparent in the following description. More particularly, in the preferred embodiment of the present invention, web-browsing engine 230 is capable of rendering various web page data (DHTML, XML, etc.) to an output device on the client side. In general, web-browsing engine 230 includes interpreters that render web page data to an output device, for example, a graphics display, sound, printer, etc. Typically, computer hardware 210, computer operating system 220 and web-browsing engine 230 comprise network-enabled device 120.

Computer system 200 preferably includes a web-based operating system 240 and a web-based computer application 250. Web-based operating system 240 functions as an interface between web-browsing engine 230 and web-based computer application 250. More particularly, web-based operating system 240 includes an application program interface ("API") that includes a set of components, routines, protocols and tools for building various web-based computer applications 250, all within the framework of web-browsing engine 230. The API includes components, or small programs that perform specific functions and are designed in such a way as to easily operate with other components and applications. Preferably, the components are implemented as JavaScript objects. Such objects are generally well known.

For example, the API includes various components for displaying information (*e.g.*, windows, icons, forms, buttons, etc.) in various manners and locations. The API includes components through which applications may interact with web-browsing engine 230 and computer operating system 220. The API also includes various components for receiving user input (*e.g.*, keyboard input, mouse input, etc.) from various sources and also components for responding to received information in an appropriate manner. With the API, application programmers may use the various components to develop web-based computer applications that run within the context of web-based operating system 240 more rapidly, consistently, and robustly.

While APIs are generally well known, the API of the present invention is novel in that it is preferably written in DHTML, JavaScript™, XML, etc. The API may also include various compiled portions (*e.g.*, C++, etc.) for complex or time sensitive operations. This is significant in that web-browsing engine 230 executes or interprets these other forms of web data into an HTML web page that is then displayed. In effect, this enables various functions of web-based computer application 250 to be handled on network-enabled device 120 (*i.e.*, client-side) rather than server 130 (*i.e.*, server-side). This is discussed in further detail below.

Computer system 200 also includes a user interface 260 that allows user 110 to navigate and control web-based computer application 250. Together with web-based computer application 250 and web-based operating system 240, user interface 260 preferably operates in a manner similar to a user interface associated with a conventional client-side computer application of a similar type. According to the present invention, a user interface of a server-side, or web-based, version of a word processor preferably operates in a manner similar to that of a conventional, local version of a word processor. Computer applications with similar user interfaces are often said to have a similar "look and feel." While having a user interface for web-based computer application 250 that is similar to that of a conventional client-side application may speed the migration of users 110 to web-based computer applications 250, those skilled in the art would recognize that this is not necessary.

The present invention enables the development, implementation, and deployment of web-based operating system 240 and web-based computer application 250. A web-based application development environment is provided that enables developers to develop web-based computer applications 250 using a component-based development model. Developers reference components, and component library files in their web-based computer application code. Component library files that include the referenced components are loaded into web-based operating system 240 the first time they are needed. Since the component library files are only loaded when they are actually needed, web-based operating system 240 and web-based computer application 250 downloads only that code that is needed for web-based operating system 240 and web-based computer application 250 to execute on network-enabled device 120.

The present invention enables the development and implementation of web-based operating system 240 and web-based computer application 250 by providing a top-level page that executes within the context of web-browsing engine 230. The top-level page is essentially a sophisticated web application that includes web page data (DHTML, XML, JavaScript, VBScript, etc.) that is processed by web-browsing engine 230. The basic functionality of web-based operating system 240 is included in the top-level page. Additional functionality for web-based operating system 240 and web-based computer application 250 is included in library files.

Two of the primary types of library files are component library files and application library files. Component library files include component definitions and logic that allow applications running within web-browsing engine 230 to instantiate component objects. Application library files are files that include an application object. The application object identifies application program logic within the application library file and allows the application program logic to be incorporated into the top-level page so that web-browsing engine 230 may execute the application. Developers of web-based computer application 250 write the code in the form of application library files.

The top-level page accesses the library files through the library manager. The library manager is a set of functions and objects, in the top-level page that loads the components and application program logic from library files into the top-level page. Once the components and application program logic are loaded into the top-level page, the web application is executed by web-browsing engine 230.

The operation of web-based operating system 240 and web-based computer application 250 begins when user 110 downloads the top-level page to web-browsing engine 230 running on network-enabled device 120. Import statements in the top-level page identify core component libraries that provide components and functionality that the top-level page requires to operate as web-based operating system 240. The library manager reads the import statements, identifies the location of the core component libraries and loads them into the top-level page. Preferably, the library manager creates a new context in the top-level page and loads the components from the library file into the context. Preferably, the new context is a frame element within the top-level page, such as a frame, iframe, layer, etc. The library manager then maps the identity of the new context (e.g., frame element) to the components and identity of the library files loaded into the new context. This mapping creates a way for the library manager to determine that a particular library file has already been loaded, and map references to a component into the context that includes that component.

After the core component libraries have been loaded, the top-level page is ready to load and execute an application library file. The application library file provides the program logic to function as an application within the top-level page and web-browsing engine 230. Preferably, the top-level page contains a reference to an application library, such as a URL address, or other information identifying an application. Alternately, user 110 may provide the URL address, or surf to the location of web-based computer application 250 according to the present invention. Regardless of the source of the reference to the application library file, the library manager begins the process of running the application within the context of web-browsing engine 230 by accessing the application library file.

The library manager first retrieves the application library file, and then begins the process of loading the application library file into the top-level page. As part of the loading process, the library manager first determines if the application library file includes any import statements that identify other component libraries. Import statements identify component library files and indicate that the application library file includes program logic that instantiates components from those component library files. If the application library includes import statements identifying other component libraries, the library manager loads the component libraries in the same manner as the core component libraries, described above. After all component libraries associated with the application library have been loaded into the top-level page, the library manager begins loading the application library into a new context within the top-level page.

Preferably, the application library file program logic uses components from the loaded component libraries to implement the functions of the application. Examples of the types of components that an application library might use are components to draw windows, components to provide menu bars, components to implement business logic, etc. Within the application library file, components are referred to by the component library file name and component name, (e.g., LibraryFileName.ComponentName). This method of referring to a component, however, is not preferred within the top-level page, since component references typically need to identify the top-level page context within which the component is defined (e.g., ContextName.ComponentName). The identity of the top-level page context was defined when the library manager loaded the component library file into the top-level page. Within the top-level page, component references from within the new context identity that includes the application program logic should refer to the context of the component to which they refer. As such, library manager replaces the component library file name portion of the component reference with the identity of the context that the component library file was loaded into.

After the context identity component file name swap has been completed, and the program logic of the application library file has been loaded into a new context, the web-browsing engine 230 executes the new application in the top-level page, thereby rendering the application functionality of the application library file to user 110 at network-enabled device 120.

FIG. 3 illustrates an exemplary embodiment of the present invention. Web application client 300 is the software installed on network-enabled device 120 that enables user 110 to interact with web-based computer application 250 via user interface 260. Web application client 300 includes the software necessary to implement the client-side of a network-based application, namely web-browsing engine 230 and operating system and application layer 310. Preferably, network-enabled device 120 already includes a web-browsing engine 230. Accordingly, in the present invention operating system and application layer 310 or portions thereof are downloaded from a server, such as server 130, to network-enabled device 120 when user 110 chooses to use a web-based application. Alternatively, web-browsing engine 230 may be downloaded with operating system and application layer 310. One reason it may be preferable to download web-browsing engine 230 with operating system and application layer 310 is that the particular web-browsing engine on network-enabled device 120 is incompatible with operating system and application layer 310.

The various elements of operating system and application layer 310 provide the functionality of web-based operating system 240 and web-based computer application 250. Although component and API based systems often blur the line of demarcation between operating system and application, generally, top-level page 320, library manager 330, core component library 340, and API library 350 function as web-based operating system 240. Similarly, application library 360 and additional component library 370 function as web-based computer application 250.

The elements of operating system and application layer 310 are loaded into the browser context of web-browsing engine 230, and are executed, or interpreted, in a manner consistent with other web documents. Moreover, the components of operating system and application layer

310 include program code, such as HTML, DHTML, JavaScript^{TM}, VBScript^{TM}, XML, etc., that executes within the context of web-browsing engine 230.

Top-level page 320 is the main application page, or document, into which other web-based application code is loaded. When user 110 accesses server 130 to download web-based operating system 240 and web-based computer application 250, top-level page 320 is the first page transmitted to network-enabled device 120. Top-level page 320 may include HTML, DHTML, JavaScript, VBScript, XML, and any other data that web-browsing engine 230 may process in order to render web-based computer application 250 to user 110. Top-level page 320 may also include references to other pages, such as script files, HTML source files, etc., which web-browsing engine 230 incorporates when top-level page 320 is loaded.

Top-level page 320 provides the core functionality for web-based operating system 240. Preferably, the core functionality is implemented as JavaScript functions, objects, constructors and prototypes, collectively referred to herein as JavaScript program elements. Constructors and prototypes are JavaScript programming elements that define objects that may be instantiated or used as a basis for a type of object oriented class-inheritance. The JavaScript program elements are made available in top-level page 320 to other portions of the web-based operating system 240 and web-based computer application 250 to form the basis of an API.

For example, the core functionality provided by top-level page 320 includes event handlers, core system objects, and library manager 330. Event handler JavaScript program elements handle events raised by web-browsing engine 230, and may be incorporated to handle user input and events raised by other portions of web-based computer application 250. System objects are JavaScript program elements for managing the execution, resource management and administration of web-based operating system 240. System objects may include, for example, objects to run portions of an application within top-level page 320, trigger an event of an object, return the location and dimensions of windows, etc.

Top-level page 320 includes library manager 330, which provides the functionality to load and reload core component library 340, API library 350, application library 360, and additional component library 370 (i.e., library files 340-370). It should be noted that although library files 340-370 are shown as discrete elements for convenience, in actual implementation, the present invention contemplates any of library files 340-370 may include a number of library files, as would be apparent. Library files 340-370 include components and program logic that are loaded into top-level page 320 to add additional functionality to operating system and application layer 310. For example, core component library 340 includes the core API components for web-based operating system 240. API library 350 includes components available for development of applications, such as web-based computer application 250. Application library 360 includes components and program logic to provide application functionality to top-level page 320. Application library 360 includes, for example, application objects that may be executed within the context of web-browsing engine 230 once the application library has been incorporated into top-level page 320. Additional component library 370 includes components that may be loaded by library manager 330 into top-level page 320 for added functionality. Additional component library 370 may include, for example, components developed by third-party developers that may be incorporated into applications in application library 360.

FIG. 4 illustrates flowchart 400, the process of running a web-based application according to the present invention. The process of running web-based computer application 250 within the context of web-browsing engine 230 begins in step 410. In step 410, user 110 accesses top-level page 320 at server 130 and loads it into web-browsing engine 230 at network-enabled device 120. The user may access top-level page 320 in a number of ways, including for example, accessing it via the Internet, an intranet, a wireless network, via a locally cached copy on network-enabled device 120, etc. After top-level page 320 has been loaded, web-browsing engine 230 begins processing the script, HTML, and/or other data, to cause the implementation of steps 410-460 in the process of flowchart 400.

In step 420, library manager 330 loads core component library 340 into top-level page 320. Loading creates a new context in top-level page 320 and loads the components and JavaScript program elements and other data from a library file into the new context. Loading core component library 340 incorporates the HTML, script, JavaScript program elements, and any other data of core component library 340 into top-level page 320. It should be noted that although step 420 specifies loading core component library 340, library files from API library 350, or application library 360 may also be loaded in step 420 as would be apparent. The process of loading library files into top-level page 320 is described in further detail below, in conjunction with FIG. 6. Generally, however, import statements in top-level page 320 identify library files, such as core component library 340, to be loaded. Import statements in top-level page 320, and library files in general identify library files to be loaded.

Library manager 330 retrieves the library file, and creates a context in top-level page 320 into which the data from the library file is loaded, such as core component library 340 in step 420. An example of such a context is an inline floating frame ("iframe") or Netscape^{TM} layer. Both iframes and layers are separate elements within top-level page 320 that allow JavaScript program elements or other data to be inserted into top-level page 320 and may be executed as applications by web-browsing engine 230. Library manager 330 creates the context in top-level page 320, using a script, or command executed by web-browsing engine 230. Examples of the types of script commands that allow the library manager 330 to create the context include the insertAdjacentHTML or new Layer() commands. After the library manager 330 creates the context, the JavaScript program elements and data from core component library 340 is inserted into the new context. After library manager 330 loads core component library 340 in step 420, the process of flowchart 400 continues at step 430.

In step 430, top-level page 320 receives a reference to application library 360. An application library is a library file that instantiates an application object. An application object is a JavaScript program element that identifies an application and it's associated program logic within a library file. Top-level page 320 may receive the reference to application library 360 in a number of ways, including a URL address, or other information identifying application library 360 included in top-level page 320 or another library file. After top-level page 320 receives the reference to application library 360 in step 430, the process of flowchart 400 continues in strep 440. In step 440, library manager 330 loads application library 360 into a new context within top-level page 320, as described above. After step 440, the process of flowchart 400 continues in step 450.

In step 450, library manager 330 loads libraries referenced by application library 360 loaded in step 440. Application library 360 references other libraries via import statements. Import statements cause library manager 330 to load the referenced libraries. In step 450, library manager 330 loads any library files referenced by application library 360. Once the referenced library files are loaded in step 450, the process of flowchart 400 continues in step 460.

In step 460, web-browsing engine 230 executes the application loaded in step 440. Library manager 330 has incorporated new application data, such as JavaScript program elements, components and other data into top-level page 320. The new application data is included in new context elements, created by library manager 330. Web-browsing engine 230 executes the application data within the new contexts, and renders the application at network-enabled device 120 to user 110.

FIG. 5 illustrates a file according to the present invention. Library file 500 includes import library header 510, components 520, export statements 530, application object 540, and application logic 550. It should be noted that library file 500 is depicted as including elements 510-550 for illustrative purposes only, and in actual implementation, library files may include as many or as few of elements 510-550 as would be apparent. Library file 500 may include import library header 510. Import library header 510 identifies additional library files to be loaded when library file 500 is loaded. Although import library header 510 may reference any type of library file, for purposes of explanation, library files referenced in import library header 510 will be referred to as "sub-library files." Sub-library files are loaded by library manager 330 as described above.

Library file 500 may include components 520. Components 520 provide functionality to web-based operating system 240 and web-based computer application 250. Preferably, a component is any programmatic element in library file 500, such as JavaScript program elements, objects, constructors, prototypes, methods, properties, events, etc.

Library file 500 may also include export statements 530. Export statements identify particular components of components 520 that are made available to library files that import library file 500. Consider, for example, the case in which library file 500 declares which sub-library files it needs for proper execution in import library header 510. When library file 500 is loaded, library manager 330 loads the sub-libraries identified in import library header 510. The sub-libraries make their components available to library file 500 through a process called exporting. If a sub-library file is defined with a component that is exported through an export statement in the sub-library, library manager 330 incorporates the component's functionality in the program logic of library file 500. When web-browsing engine 230 runs the application, library file 500 may properly reference and use the exported sub-library components.

Library file 500 may also include application object 540, which is an instance of an object defined in top-level page 320. Application object 540 identifies library file 500 as an application library file. Application library files contain program logic and may execute as application within a new context, such as an iframe or layer, within top-level page 320. For example, an application library file may provide the functionality for a window of a word processor, a menu bar, a spreadsheet program, etc. Application logic 550 is the program logic that implements the application library file. When top-level page 320 references the application library file, preferably via a URL address, the file is loaded and the function identified by the application object is executed by web-browsing engine 230.

FIG. 6 further illustrates the process of loading library file 500 according to the present invention. The process of flowchart 600 begins in step 602, with the occurrence of a load library event. A load library event identifies a library file, described herein as library file 500 for illustrative purposes, and is any event that causes library manager 330 to load library file 500. Examples of load library events are sub-libraries identified by import library header 510 in library file 500, a reference to the URL address of an application library file within top-level page 320, an import library statement within top-level page 320, etc. After the occurrence of a load library event, the process of flowchart 600 continues in step 604.

In step 604, library manager 330 determines if library file 500 has already been loaded into a new context in top-level page 320. Library manager 330 loads library files by creating a new context within top-level page 320, such as an iframe or layer, and loads the library file into it. Library manager 330 then writes to a data structure, such as an array, mapping the newly created context to the particular library file loaded. In step 604, library manager 330 checks the data structure to determine if library file 500 has yet been loaded. If library manager 330 determines that library file 500 has not yet been loaded, the process of flowchart 600 continues in step 608.

In step 608, library manager 330 accesses and retrieves library file 500 from its location. The method by which library manager 330 accesses library file 500 is dependent upon the nature of the load library event and library file 500. For example, the locations of core component library 340 and API library 350 should be identified in top-level page 320, since these libraries form part of the core of web-based operating system 240. Application library 360, and additional component library 370, on the other hand, may originate from third party developers, and their location may not be known before the load library event occurs.

Preferably, top-level page 320 includes a data structure that maps an identifier for library file 500, such as the name of the file, or other uniquely identifying information, to the location of library file 500. Library file 500 may be located anywhere on network 140, server 130, network-enabled device 120, etc. In step 608, library manager 330 accesses the data structure, determines the location of, and retrieves, library file 500. In an alternate embodiment, the load library event may identify the location of library file 500, such as the URL address of library file 500. In such cases, library manager 330 accesses the location of library file 500 identified by the load library event. After library manager 330 accesses and retrieves library file 500 in step 608, the process of flowchart 600 continues in step 610. In step 610, library manager 330 creates a new context in top-level page 320 and loads library file 500 into it. In step 610, library manager 330 creates an entry in a data structure in top-level page 320 that maps the identifier of library file 500 to the new context. After step 610, the process of flowchart 600 continues in step 612.

If, on the other hand, in step 604, library manager 330 determines that library file 500 has already been loaded, the process of flowchart 600 continues in step 612. Library manager 330 determines that library file 500 has already been loaded by checking the data structure in top-level page 320 where the identifier of library file 500 is mapped to the new context. If the mapping exists, library manager 330 determines that library file 500 has already been loaded.

In step 612, library manager 330 determines if library file 500 identifies any sub-libraries to import. Library file 500 identifies sub-library files to import in import library header 510, as described above. If library manager 330 determines library file 500 does not identify sub-library files in import library header 510, the process of flowchart 600 continues in step 624. If, on the other hand, library file 500 includes sub-libraries in import library header 510, the process of flowchart 600 continues in step 616. It should be noted, however, that the process for importing a sub-library is the same as that for any library. When library manager 330 imports a sub-library file, the sub-library file is loaded with the process of flowchart 600. Using the process of flowchart 600 to load sub-library files ensures recursive library file loading. For example, sub-library files identified in import library header 510 may themselves import additional sub-library files (i.e., sub-sub-library files). As such, the process of flowchart 600 ensures that all sub-library files identified in import library header 510 are loaded. If library manager 330 determines that library file 500 includes an import library header 510 referencing sub-library files, the process of flowchart 600 continues in step 616.

In step 616, library manager 330 determines whether the sub-library identified by import library header 510 has yet been loaded into a context within top-level page 320. Library manager 330 checks a data structure in top-level page 320 to determine if the sub-library file identified in import library header 510 is associated with a context within top-level page 320. If the sub-library file has not yet been loaded, the process of flowchart 600 continues in step 618. In step 618, library manager 330 accesses the location of, and retrieves the sub-library file. The location of the sub-library file may be stored in a data structure of top-level page 320, or alternatively, passed to library manager 330 with the import statement in import library header 510. After library manager 330 accesses and retrieves the sub-library in step 618, the process of flowchart 600 continues in step 620. In step 620, library manager 330 loads the sub-library file, as described above. After library manager 330 loads the sub-library file in step 620, the process of flowchart 600 continues in step 622.

If, on the other hand, library manager 330 determines, in step 616, that the sub-library file identified in import library header 510 has already been loaded, the process of flowchart 600 continues in step 622. In step 622, library manager 330 determines if import library header 510 identifies additional sub-library files to be loaded. If import library header 510 identifies additional sub-library files, the process of flowchart 600 continues at step 616, where steps 616-622 are iterated until all of the sub-library files have been loaded. If, on the other hand, library manager 330 determines there are no additional sub-library files to load, the process of flowchart 600 continues in step 624.

In step 624, library manager 330 adds components and JavaScript program elements exported by sub-library files to library file 500. The exported components are added to library file 500 for purposes of code reuse and rapid application development, as are the goals of the typical component-based development model. For each component exported by a sub-library, library manager 330 creates an entry in a data structure, such as an array, in top-level page 320. The entry identifies the sub-library that includes the component, a component identifier (i.e., name), and the context identifier, such as the name of the iframe or layer, into which the sub-library was loaded. When library file 500 is loaded into a new context, library manager 330 replaces the references to sub-library components with the context identifier for the sub-library and component name. This, effectively, maps the components from the sub-library context into the context of library file 500 via conventional inter-frame scripting.

In conventional JavaScript inter-frame scripting, program logic in a first context is able access the components and objects within a second context. The program logic in the first context, however, must explicitly identify the second context when instantiating objects defined within the second context. Because the present invention loads library files dynamically, the identities of contexts that include components and JavaScript program elements are not fixed, and therefore cannot be explicitly identified by program logic of application library files.

Library manager 330 removes the requirement that library file 500 explicitly refer to the context of a sub-library by maintaining a map between components, and the identity of library file contexts that include them. When a new library file is loaded, library manager 330 replaces references to sub-library components in the new library file with the identities of the contexts in which the sub-library components are loaded. When the new library file is loaded into a context, therefore, it explicitly identifies the context of the sub-library that includes the components.

In summary, the present invention provides a component-based development model for the development and implementation of web-based operating system 240 and web-based computer application 250. Components from core component library 340 and API library 350 are loaded into top-level page 320 by library manager 330 for use by web-based operating system 240. Library manager 330 loads application library 360 into top-level page 320, and also loads any additional component library 370 referenced by application library 360. Once application library 360 and any related component libraries have been loaded into top-level page 320, web-browsing engine 230 executes top-level page 320 to provide web-based operating system 240 and web-based computer application 250 functionality to user 110 at network-enabled device 120.

Typically, applications executing within the context of web-browsing engine 230 are not persistent. Top-level page 320 and the associated component and application library files are preferably downloaded via network 140 to network-enabled device 120 and execute within the context of web-browsing engine 230. When user 110 exits the browser application or reboots network-enabled device 120, the downloaded files are usually discarded. Upon restart, or when reloading web-browsing engine 230, user 110 accesses top-level page 320 and the associated application and component library files by, once again, accessing them via network 140. The operation of web-based operating system 240 and web-based computer application 250, therefore is not persistent. This lack of persistence of the web-based operating system and web-based computer applications provides for a payment model based on the usage of particular web-based computer applications, since user 110 must have access to download for each use. Pay per use charges with web-based computer applications are a more desirable alternative to the pay per instance of conventional locally loaded computer application.

While the invention has been described in detail and with reference to specific embodiments thereof, it will be apparent to one skilled in the art that various changes and modifications can be made therein without departing from the spirit and scope thereof. Thus, it is intended that the present invention cover the modifications and variations of this invention provided they come within the scope of the appended claims and their equivalents.

What is claimed is:

1. A method for executing a web-based application within a browser, comprising:

loading a top-level page, said top-level page referencing a first library file;

loading said first library file, wherein said first library file includes an application program interface component;

loading a second library file, wherein said second library file references said application program interface component; and

executing said application program interface component referenced by said second library file.

2. The method of claim 1, wherein said top-level page provides an execution context for said first library file and for said second library file.

3. The method of claim 1, wherein said top-level page includes a library manager for loading library files.

4. The method of claim 1, wherein said loading said first library file further comprises loading said first library file into a new context within said top-level page.

5. The method of claim 4, wherein said new context is an iframe.

6. The method of claim 4, wherein said new context is a layer.

7. The method of claim 1, wherein said loading said second library file further comprises loading said second library file into a new context within said top-level page.

8. The method of claim 7, wherein said new context is a new iframe.

9. The method of claim 7, wherein said new context is a new layer.

10. The method of claim 1, further comprising loading a sub-library file in response to a reference to said sub-library file in said first library file.

11. The method of claim 1, further comprising loading a sub-library file in response to a reference to said sub-library file in said second library file.

12. The method of claim 1, wherein said second library file is loaded in response to user interaction with the browser.

13. A method for implementing a web-based application program interface, comprising:

loading a top-level page into a web-browser, said top-level page adapted to be processed by the web-browser;

receiving a request to load a first library file, said first library file including an application program interface component;

determining if said first library file has been loaded;

loading said first library file into a new context in said top-level page if said first library file has not been loaded;

determining if said first library file references a second library file;

determining if said second library file has been loaded; and

loading said second library file into a new context in said top-level page if said second library file has not been loaded.

14. The method of claim 13, wherein said top-level page includes a library manager for loading library files.

15. The method of claim 13, wherein said request is included in said top-level page.

16. The method of claim 13, wherein said request is generated by user interaction with the web-browser.

17. The method of claim 13, wherein said request is generated by a web-based application executing in the web-browser.

18. The method of claim 13, wherein said new context is an iframe.

19. The method of claim 13, wherein said new context is a layer.

20. A computer program product for enabling a processor in a computer system to implement a system for executing a web-based application within a browser, said computer program product comprising:

a computer usable medium having computer readable program code means embodied in said computer usable medium for causing a program to execute on the computer system, said computer readable program code means comprising:

means for enabling the computer system to load a top-level page, said top-level page referencing a first library file;

means for enabling the computer system to load said first library file, wherein said first library file includes an application program interface component;

means for enabling the computer system to load a second library file, wherein said second library file references said application program interface component; and

means for enabling the computer system to execute said application program interface component referenced by said second library file.

21. The computer program product of claim 20, wherein said top-level page provides an execution context for said first library file and for said second library file.

22. The computer program product of claim 20, wherein said top-level page includes a library manager for loading library files.

23. The computer program product of claim 20, wherein said means for enabling the computer system to load said first library file further comprises means for enabling the computer system to load said first library file into a new context within said top-level page.

24. The computer program product of claim 23, wherein said new context is an iframe.

25. The computer program product of claim 23, wherein said new context is a layer.

26. The computer program product of claim 20, wherein said means for enabling the computer system to load said second library file further comprises said means for enabling the computer system to load said second library file into a new context within said top-level page.

27. The computer program product of claim 26, wherein said new context is a new iframe.

28. The computer program product of claim 26, wherein said new context is a new layer.

29. The computer program product of claim 20, further comprising means for enabling the computer system to load a sub-library file in response to a reference to said sub-library in said first library file.

30. The computer program product of claim 20, further comprising means for enabling the computer system to load a sub-library file in response to a reference to said sub-library in said second library file.

31. The computer program product of claim 26, wherein said second library file is loaded in response to user interaction with the browser.

32. A computer program product for enabling a processor in a computer system to implement a system for executing a web-based application within a browser, comprising:

a computer usable medium having computer readable program code means embodied in said computer usable medium for causing a program to execute on the computer system, said computer readable program code means comprising:

means for enabling the computer system to load a top-level page, said top-level page referencing a first library file;

33. A computer program product for enabling a processor in a computer system to implement a system for implementing a web-based application program interface, said computer program product comprising:

a computer usable medium having computer readable program code means embodied in said computer usable medium for causing a program to execute on the computer system, said computer readable program code means comprising:

means for enabling the computer system to load a top-level page into a web-browser, said top-level page adapted to be processed by the web-browser;

means for enabling the computer system to receive a request to load a first library file, said first library file including an application program interface component;

means for enabling the computer system to determine if said first library file has been loaded;

means for enabling the computer system to load said first library file into a new context in said top-level page if said first library file has not been loaded;

means for enabling the computer system to determine if said first library file references a second library file;

means for enabling the computer system to determine if said second library file has been loaded; and

means for enabling the computer system to load said second library file into a new context in said top-level page if said second library file has not been loaded.

34. The computer program product of claim 33, wherein said top-level page includes a library manager for loading library files.

35. The computer program product of claim 33, wherein said request is included in said top-level page.

36. The computer program product of claim 33, wherein said request is generated by user interaction with the web-browser.

37. The computer program product of claim 33, wherein said request is generated by web-based application executing in the web-browser.

38. The computer program product of claim 33, wherein said new context is an iframe.

39. The computer program product of claim 33, wherein said new context is a layer.

Abstract of the Invention

A system and method for providing a web-based operating system is disclosed. A web-based operating system is downloaded from a server to a network-enabled device. The web-based operating system provides a platform from which to run web-based computer applications. Web-based computer applications are downloaded from the server onto the network-enabled device, and executed in conjunction with the web-based operating system. Those portions of the web-based operating system software and web-based computer application software are downloaded to the network-enabled device as needed. A component-based environment for the development and deployment of a web-based operating system and web-based computer applications is also provided. Components are deployed within library files as part of a component-based development model. The web-based operating system downloads those library files needed to execute the web-based computer applications.

59657 v1/RE
1@1501!.DOC
100900/1057

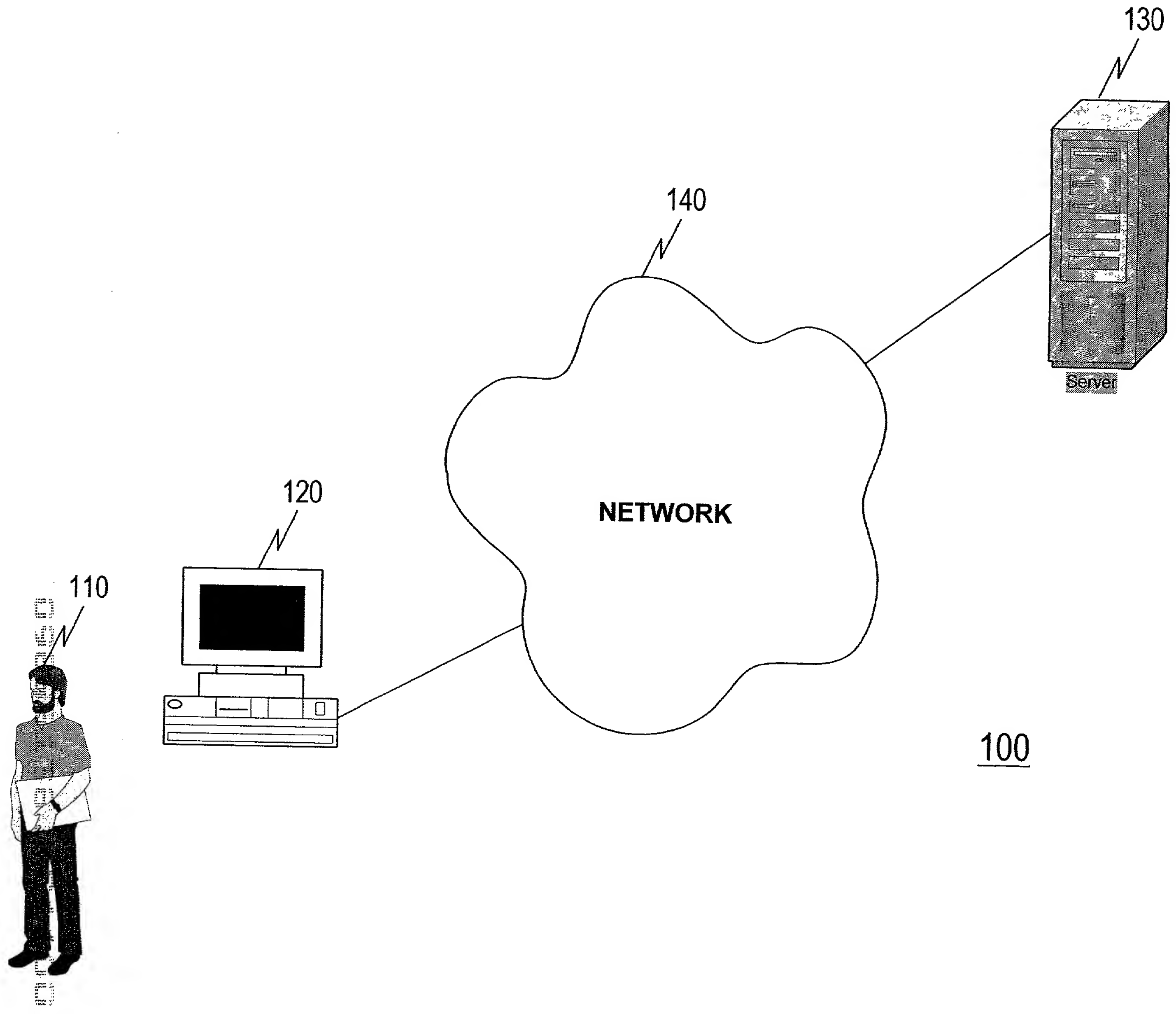


FIG. 1

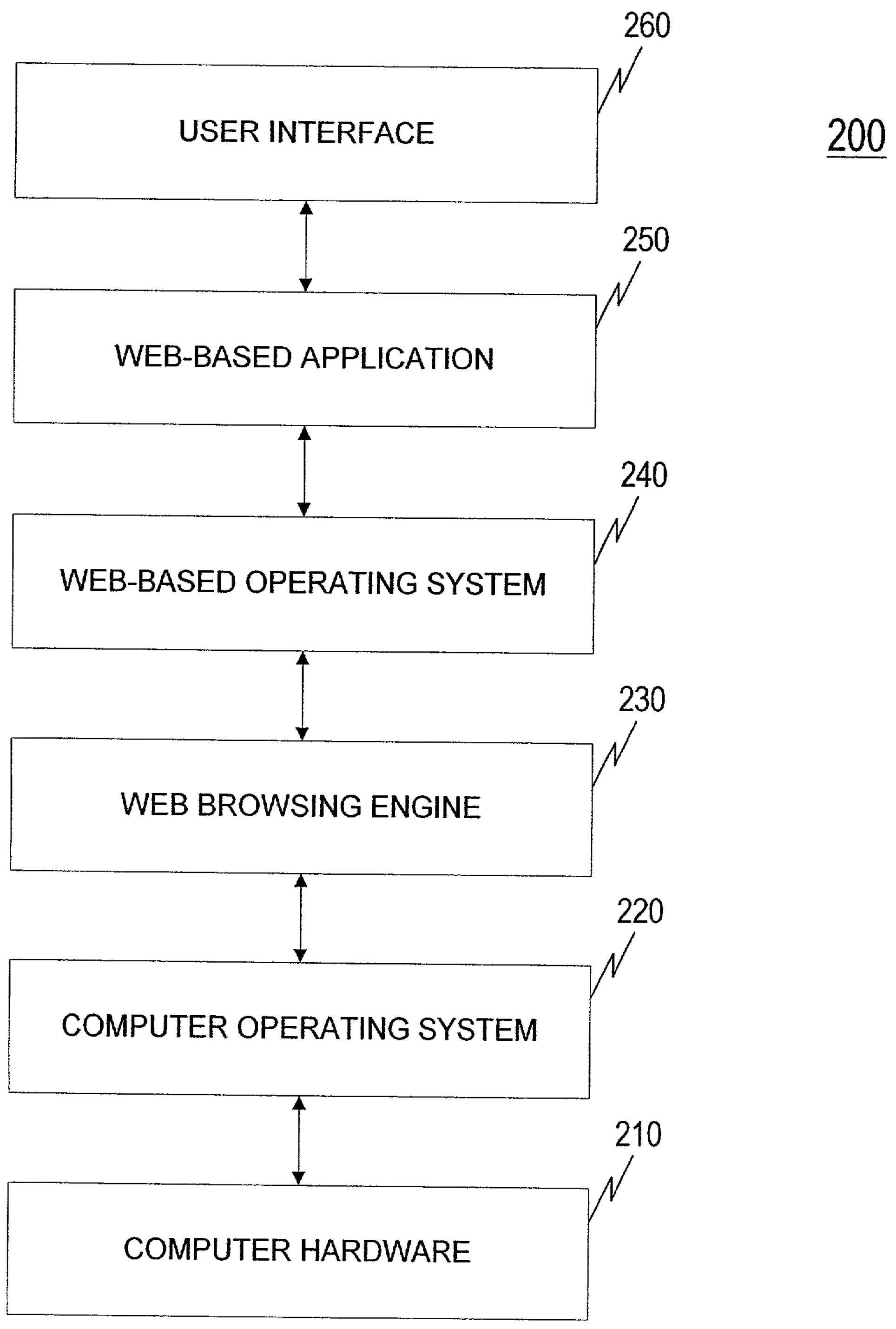
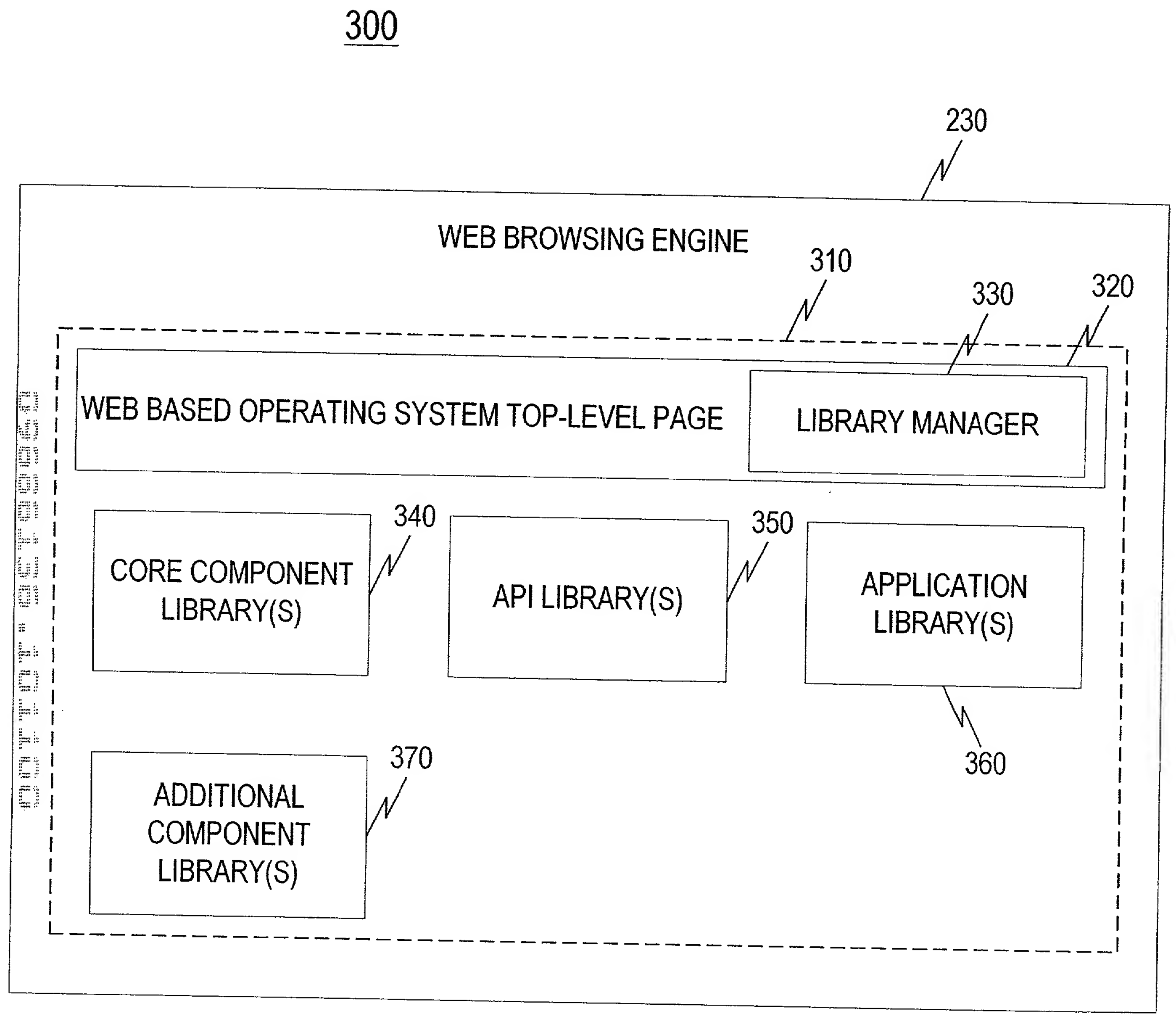

260
USER INTERFACE
200
250
WEB-BASED APPLICATION
240
WEB-BASED OPERATING SYSTEM
230
WEB BROWSING ENGINE
220
COMPUTER OPERATING SYSTEM
210
COMPUTER HARDWARE

FIG. 2



300
230
WEB BROWSING ENGINE
310
330
320
WEB BASED OPERATING SYSTEM TOP-LEVEL PAGE
LIBRARY MANAGER
340
CORE COMPONENT LIBRARY(S)
350
API LIBRARY(S)
APPLICATION LIBRARY(S)
360
370
ADDITIONAL COMPONENT LIBRARY(S)

FIG. 3

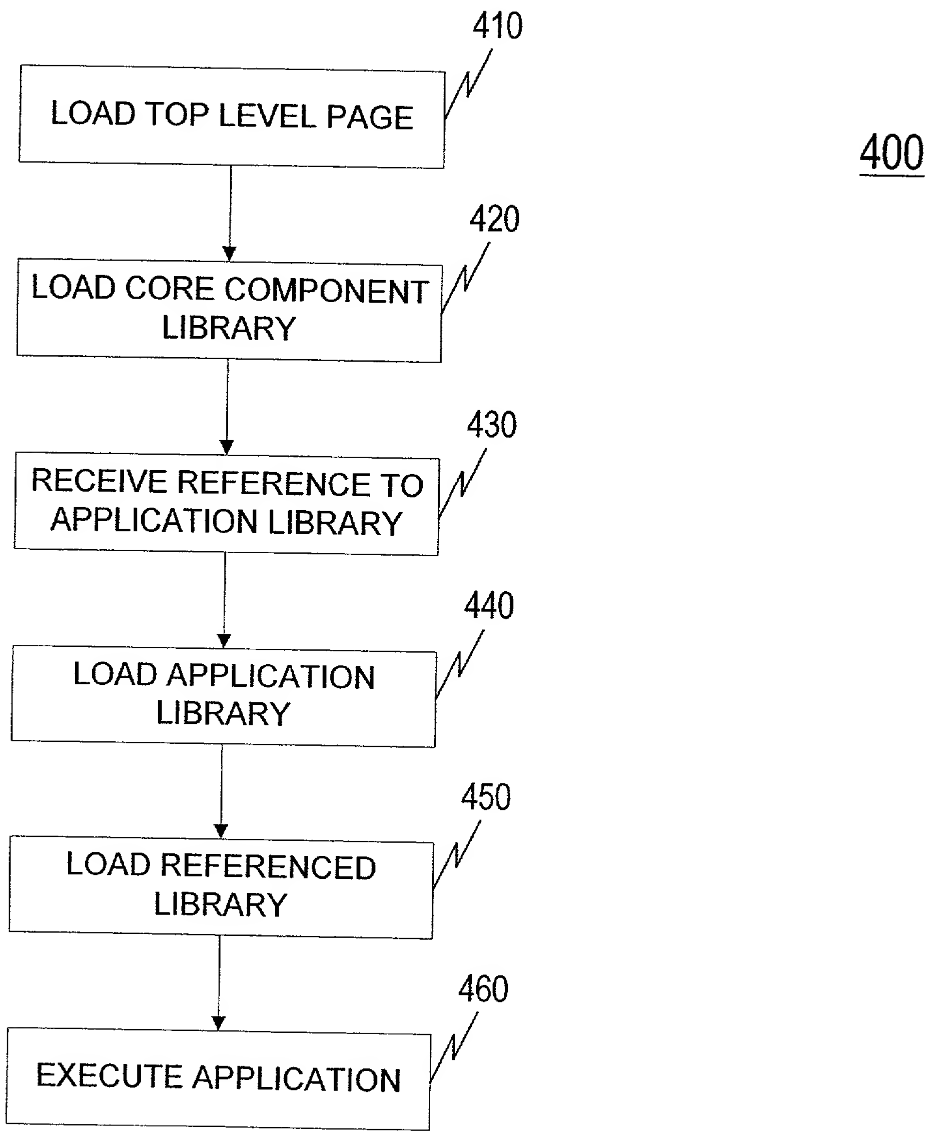

410
LOAD TOP LEVEL PAGE
420
LOAD CORE COMPONENT LIBRARY
430
RECEIVE REFERENCE TO APPLICATION LIBRARY
440
LOAD APPLICATION LIBRARY
450
LOAD REFERENCED LIBRARY
460
EXECUTE APPLICATION
400

FIG. 4

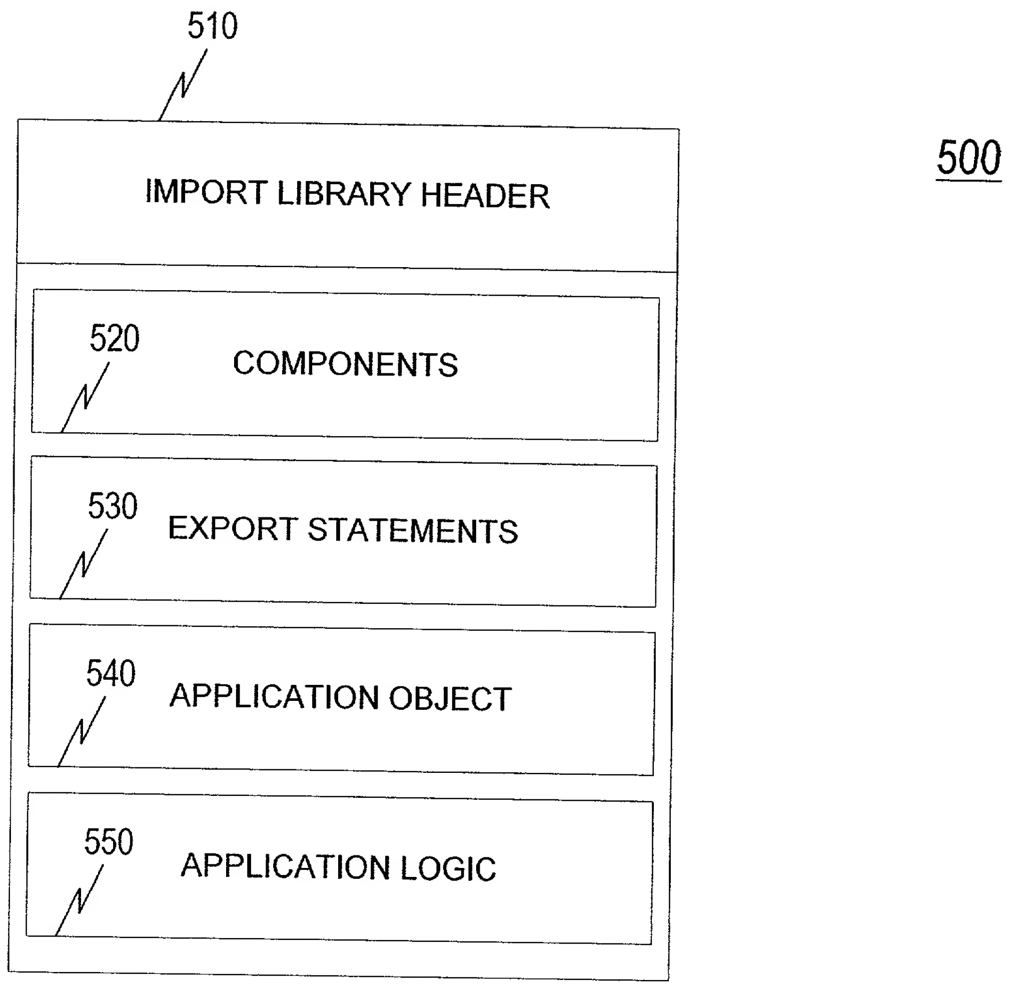

510
IMPORT LIBRARY HEADER
500
520
COMPONENTS
530
EXPORT STATEMENTS
540
APPLICATION OBJECT
550
APPLICATION LOGIC

FIG. 5

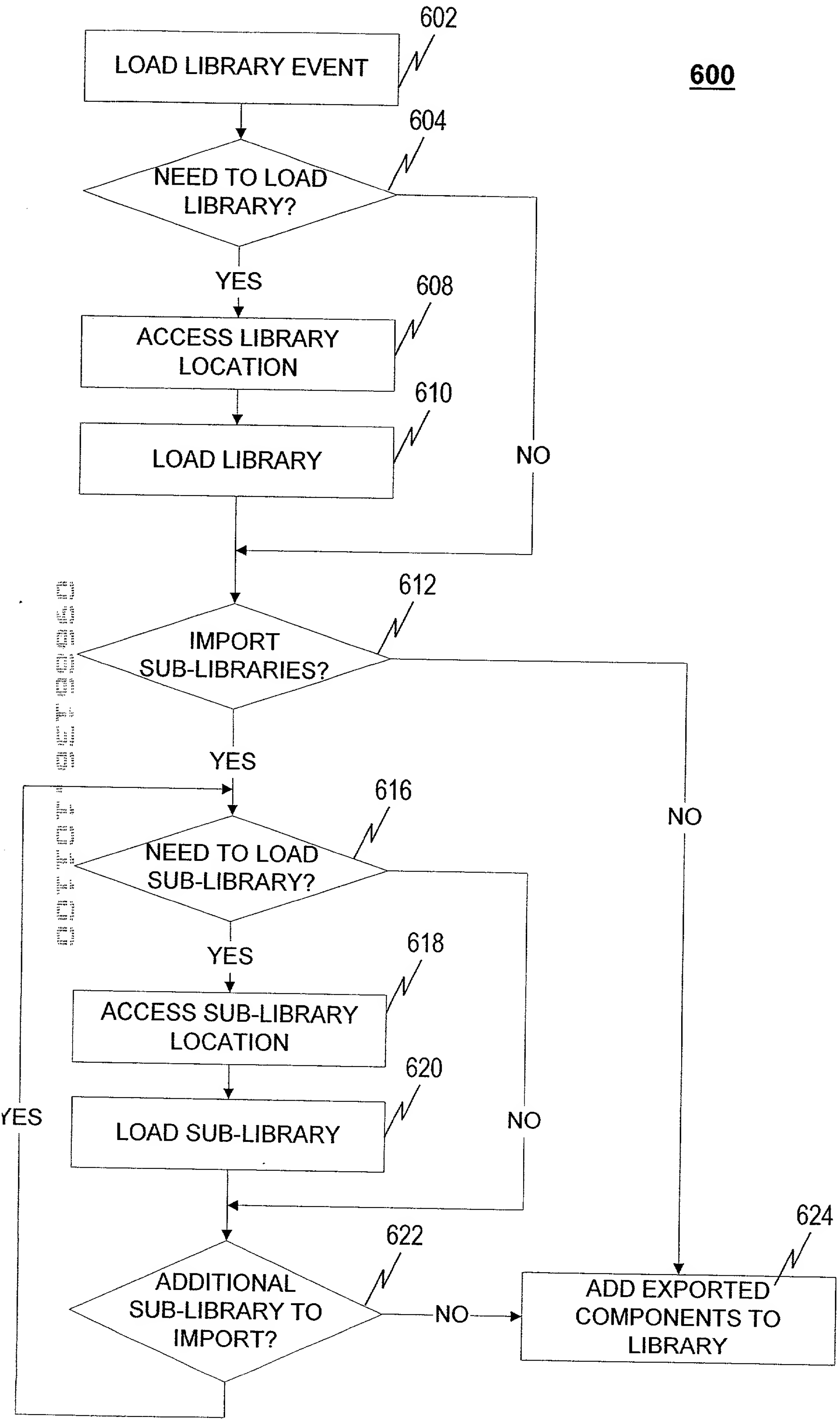

600
602
LOAD LIBRARY EVENT
604
NEED TO LOAD LIBRARY?
YES
NO
608
ACCESS LIBRARY LOCATION
610
LOAD LIBRARY
612
IMPORT SUB-LIBRARIES?
YES
NO
616
NEED TO LOAD SUB-LIBRARY?
YES
NO
618
ACCESS SUB-LIBRARY LOCATION
620
LOAD SUB-LIBRARY
622
ADDITIONAL SUB-LIBRARY TO IMPORT?
YES
NO
624
ADD EXPORTED COMPONENTS TO LIBRARY

FIG. 6

Attorney Docket No: HYPE-001/01US **PATENT**

DECLARATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

<u>SYSTEM AND METHOD FOR PROVIDING A WEB-BASED OPERATING SYSTEM</u>

the specification of which is attached hereto;

I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above;

I acknowledge the duty to disclose to the U.S. Patent and Trademark Office all information which is known to me to be material to the patentability of said invention in accordance with 37 C.F.R. §1.56;

I do not know and do not believe said invention was ever known or used in the United States of America before my or our invention thereof, or patented or described in any printed publication in any country before my or our invention thereof or more than one year prior to said application; that said invention was not in public use or on sale in the United States of America more than one year prior to said application; that said invention has not been patented or made the subject of an inventor's certificate issued before the date of said application in any country foreign to the United States of America on any application filed by me or my legal representatives or assigns more than six months prior to said application;

I hereby claim foreign priority benefits under 35 U.S.C. §119 and/or §365 of any foreign application(s) for patent, any foreign application(s) for inventor's certificate, or any PCT international application(s) designating at least one country other than the United States of America listed below; I have also identified below any foreign application(s) for patent, any foreign application(s) for inventor's certificate, or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed:

Prior Foreign Application(s)

COUNTRY/INTERNATIONAL	APPLICATION NUMBER	DATE OF FILING (day, month, year)	PRIORITY CLAIMED
			YES NO

I hereby claim the benefit under 35 U.S.C. §119(e) of any United States provisional application(s) listed below:

60/186,304	March 1, 2000
(Application Number)	(Filing Date) (day, month, year)

I hereby claim the benefit under 35 U.S.C. §120 and/or §365 of any United States application(s) or of any international application(s) designating the United States of America that is/are listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior application(s) in the manner provided by the first paragraph of 35 U.S.C. §112, I acknowledge the duty to disclose to the U.S. Patent and Trademark Office all information known to me to be material to patentability as defined in 37 C.F.R. §1.56 which became available between the filing date(s) of the prior application(s) and the national or PCT international filing date of this application:

Prior U.S. Application(s) or PCT International Applications Designating the U.S. for benefit under 35 U.S.C. §120

U.S. APPLICATIONS			**STATUS (*check one*)**		
U.S. APPLICATION NO.	U.S. FILING DATE (day, month, year)		Pending	Patented	Abandoned
PCT APPLICATIONS DESIGNATING THE U.S.					
PCT APPLICATION NO.	PCT FILING DATE (day, month, year)	U.S. APPLICATION NOS. (if any)			

As a named inventor, I hereby appoint the registered practitioners of Cooley Godward LLP included in the Customer Number provided below to prosecute this application and to transact all business in the U.S. Patent and Trademark Office connected therewith; I further direct that correspondence concerning this application be directed to:

COOLEY GODWARD LLP
Attention: Patent Group
One Freedom Square - Reston Town Center
11951 Freedom Drive
Reston, Virginia 20190-5601
Tel: (703) 456-8000 - Fax: (703) 456-8100
CUSTOMER NUMBER: **022903**

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of first inventor: **Fredrik Malmer**

Inventor's signature [illegible] Date 09/21/2000
Residence: Odengatan 140 Alingsas 441 51 Sweden
Citizen of: Sweden
Post Office Address: Odengatan 140 Alingsas 441 51 Sweden

Full name of second inventor: **Erik Viktor Arvidsson**

Inventor's signature ________________________ Date ____________________
Residence: Betselvagen 9 55447 Jonkoping Sweden
Citizen of: Sweden
Post Office Address: Betselvagen 9 55447 Jonkoping Sweden

Full name of third inventor: **Drew E. Morris**

Inventor's signature ________________________ Date ____________________
Residence: 23 Timberline Drive, Alpine, NJ 07620
Citizen of: United States of America
Post Office Address: 23 Timberline Drive, Alpine, NJ 07620

Full name of fourth inventor: **Shervin Pishevar**

Inventor's signature ________________________ Date ____________________
Residence: 46 Federal Court, Gaithersburg, MD 20877
Citizen of: United States of America
Post Office Address: 46 Federal Court, Gaithersburg, MD 20877

64715 v1/RE
1DXN011.DOC

END

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of first inventor: **Frederik Malmer**

Inventor's signature ______________________ Date ________________
Residence: Odengaten 140 Alingsas 441 51 Sweden
Citizen of: Sweden
Post Office Address: Odengaten 140 Alingsas 441 51 Sweden

Full name of second inventor: **Erik Viktor Arvidsson**

Inventor's signature [illegible] Date 9/18/00
Residence: Betselvagen 9 55447 Jonkoping Sweden
Citizen of: Sweden
Post Office Address: Betselvagen 9 55447 Jonkoping Sweden

Full name of third inventor: **Drew E. Morris**

Inventor's signature [illegible] Date 10/04/00
Residence: 23 Timberline Drive, Alpine, NJ 07620
Citizen of: United States of America
Post Office Address: 23 Timberline Drive, Alpine, NJ 07620

Full name of fourth inventor: **Shervin Pishevar**

Inventor's signature [illegible] Date 10/04/00
Residence: 46 Federal Court, Gaithersburg, MD 20877
Citizen of: United States of America
Post Office Address: 46 Federal Court, Gaithersburg, MD 20877

Attorney Docket No: HYPE-001/01US **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of Frederik Malmer *et al.*

Serial No.:	Not Yet Assigned	Examiner:	Unassigned
Filed:	Herewith	Art Unit:	Unassigned

For: SYSTEM AND METHOD FOR PROVIDING A WEB-BASED OPERATING SYSTEM

Assistant Commissioner for Patents
Washington, D.C. 20231

POWER OF ATTORNEY BY ASSIGNEE
AND STATEMENT UNDER 37 C.F.R. §3.73(b)

As the assignee of record of the entire right, title, and interest in the above-identified application, the registered practitioners of Cooley Godward LLP included in the Customer Number provided below are hereby granted Power of Attorney to prosecute this application and any applications claiming priority to this application, and to transact all business in the U.S. Patent and Trademark Office connected therewith.

The assignee certifies that to the best of its knowledge and belief it is the owner of the entire right, title, and interest in and to the above-identified application as evidenced by:

[x] An assignment document, a copy of which is enclosed herewith;

[ ] An assignment previously recorded in the U.S. Patent and Trademark Office at Reel _____, Frame _____.

Please direct all telephone calls and correspondence to:

COOLEY GODWARD LLP
Attention: Patent Group
One Freedom Square - Reston Town Center
11951 Freedom Drive
Reston, Virginia 20190-5601
Tel: (703) 456-8000
Fax: (703) 456-8100

CUSTOMER NUMBER: **022903**

The undersigned (whose title is supplied below) is empowered to sign this statement on behalf of the assignee.

Date: 9/13/00 Signature: [illegible]

Name: Shervin Pishevar
Title: Chief Executive Officer
Company: WebOS, Inc.

64737 v1/RE
1DY901!.DOC

Attorney Docket No: HYPE-001/01US **PATENT**

ASSIGNMENT
(Joint)

Frederik Malmer, residing at Odengaten 140 Alingsas 441 51 Sweden; Erik Viktor Arvidsson, residing at Betselvagen 9 55447 Jonkoping Sweden; Drew E. Morris, residing at 23 Timberline Drive, Alpine, NJ 07620; and Shervin Pishevar, residing at 46 Federal Court, Gaithersburg, MD 20877 (hereinafter referred to as "Assignor") has made an invention(s) relating to certain new and useful improvements in:

SYSTEM AND METHOD FOR PROVIDING A WEB-BASED OPERATING SYSTEM

set forth in an application for Letters Patent of the United States, which is a

(1) [x] non-provisional application
 (a) [ ] having an oath or declaration executed on prior to filing of application; or
 (b) [x] to be filed herewith; or
 (a) [ ] bearing Application No. , and filed on ; and

WHEREAS, WebOS, Inc., a corporation duly organized under and pursuant to the laws of Delaware, and having its principal place of business at 10500 Little Patuxent Parkway, Suite 500, Columbia, MD 21044 (hereinafter referred to as "the Assignee"), is desirous of acquiring the entire right, title, and interest in and to said invention(s), the right to file applications on said invention(s) and the entire right, title and interest in and to any applications, including provisional applications for Letters Patent of the United States or other countries claiming priority to said application, and in and to any Letters Patent or Patents, United States or foreign, to be obtained therefor and thereon;

NOW, THEREFORE, in consideration of One Dollar ($1.00) and other good and sufficient consideration, the receipt of which is hereby acknowledged, the Assignor has sold, assigned, transferred, and set over, and by these presents does sell, assign, transfer, and set over, unto the Assignee, its successors, legal representatives, and assigns the entire right, title, and interest in and to the above-mentioned invention(s), the right to file applications on said invention(s) and the entire right, title and interest in and to any applications for Letters Patent of the United States or other countries claiming priority to said application, and any and all Letters Patent or Patents of the United States of America and all foreign countries that may be granted therefor and thereon, and in and to any and all applications claiming priority to said applications, divisions, continuations, and continuations-in-part of said applications, and reissues and extensions of said Letters Patent or Patents, and all rights under the International Convention for the Protection of Industrial Property, the same to be held and enjoyed by the Assignee, for its own use and behalf and the use and behalf of its successors, legal representatives, and assigns, to the full end of the term or terms for which Letters Patent or Patents may be granted as fully and entirely as the same would have been held and enjoyed by the Assignor had this sale and assignment not been made;

AND for the same consideration, the Assignor hereby covenants and agrees to and with the Assignee, its successors, legal representatives, and assigns, that, at the time of execution and delivery of these presents, the Assignor is the sole and lawful owner of the entire right, title, and interest in and to the invention(s) set forth in said applications and said applications, including provisional applications, above-mentioned, and that the same are unencumbered, and that the Assignor has good and full right and lawful authority to sell and convey the same in the manner herein set forth;

AND for the same consideration, the Assignor hereby covenants and agrees to and with the Assignee, its successors, legal representatives, and assigns that the Assignor will, whenever counsel of the Assignee, or the counsel of its successors, legal representatives, and assigns, shall advise that any proceeding in connection with said invention(s) or said applications for Letters Patent or Patents, or any proceeding in connection with Letters Patent or Patents for said invention(s) in any country, including interference proceedings, is lawful and desirable, or that any application claiming priority to said application, division, continuation, or continuation-in-part of any applications for Letters Patent or Patents, or any reissue or extension of any Letters Patent or Patents to be obtained thereon, is lawful and desirable, sign all papers and documents, take all lawful oaths, and do all acts necessary or required to be done for the procurement, maintenance, enforcement and defense of Letters Patent or Patents for said invention(s), without charge to the Assignee, its successors, legal representatives, and assigns, but at the cost and expense of the Assignee, its successors, legal representatives, and assigns;

AND the Assignor hereby authorizes and requests the attorneys of COOLEY GODWARD L.L.P. to insert in the spaces provided above the filing date, application number, the date of execution of the oath or declaration, and attorney docket number of said application when known;

AND the Assignor hereby requests the Commissioner of Patents to issue any and all said Letters Patent of the United States to the Assignee, as the Assignee of said invention(s) and the Letters Patent to be issued thereon for the sole use and behalf of the Assignee, its successors, legal representatives, and assigns.

Date: 09/21/2000 By: [illegible]
Fredrik Malmer

Witnesses:

Date: 09/21/2000 By: [illegible]

Date: 09/21/2000 By: [illegible]

COPY

Date: 10/04/00 By: [illegible]
Erik Viktor Arvidsson

Witnesses:

Date: 10/04/00 By: [illegible]

Date: 10/4/00 By: Shahrokh Malis

Date: 10/04/00   By: [illegible]
Drew E. Morris

State of: Maryland

County of: Anne Arundel

The preceding Assignment was acknowledged before me this 4 day of Oct 2000 by Drew Morris

[illegible]

Notary Public

My Commission Expires: June 1, 2004

Date: 10/04/00　　By: [illegible]
Shervin Pishervar

State of: Maryland

County of: Anne Arundel

The preceding Assignment was acknowledged before me this 4 day of Oct. 2000 by Shervin Pishervar

M Lynn Daye
Notary Public

My Commission Expires: 6/1/2004

64720 v1/RE
1DXS01!.DOC